DOCKET NO.: IVGP-0002 **PATENT**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: **Hiang-Swee Chiang** Confirmation No.: **2184**

Serial No.: **09/810,716** Group Art Unit: 2193

Filing Date: **March 16, 2001** Examiner: **William H. Wood**

For: **WEB APPLICATION GENERATOR**

EXPRESS MAIL LABEL NO: ER 337148997 US
DATE OF DEPOSIT: January 31, 2007

Mail Stop Appeal-Brief Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANT'S BRIEF PURSUANT TO 37 C.F.R. § 41.37

Table of Contents

Sir:

APPELLANT'S BRIEF PURSUANT TO 37 C.F.R. § 41.37

This brief is being filed in support of Appellant's appeal from the rejections of claims 2, 4-29, 31-50, 52-64, 66-78 and 162-169 dated May 17, 2006. A Notice of Appeal was filed on June 30, 2006.

1. REAL PARTY IN INTEREST

Gutenberg Printing LLC is the real party of interest by virtue of an assignment recorded October 4, 2004, at Reel 015213, Frame 0050.

2. RELATED APPEALS AND INTERFERENCES

None

3. STATUS OF CLAIMS

Claim 1	Cancelled
Claims 2	Rejected and On Appeal
Claim 3	Cancelled
Claims 4-29	Rejected and On Appeal
Claim 30	Cancelled
Claims 31-50	Rejected and On Appeal
Claim 51	Cancelled
Claims 53-64	Rejected and On Appeal
Claim 65	Cancelled
Claims 66-78	Rejected and On Appeal
Claim 79-161	Cancelled
Claims 162-169	Rejected and On Appeal
Claims 170-174	Cancelled

4. STATUS OF AMENDMENTS

The Appellant's' amendment of February 21, 2006 has been entered.

5. SUMMARY OF CLAIMED SUBJECT MATTER

The claimed subject matter is embodied in methods and apparatus for generating a web application. In accordance with the invention, input files from graphic designers may be dynamically bound with source code from web developers. Input files from a web application graphical user interface are parsed for tag type, attribute name, and attribute value. The resulting event handler is organized into web application source code.

The foregoing can be better understood with reference to exemplary claim 2, which is reproduced below with explanatory annotations to the flow charts and specification describing the flow charts:

CLAIM 2	DESCRIPTION	DRAWINGS
2. A method of generating computer code for a web application, comprising:	The method of generating computer code in accordance with the invention is shown in Figs 5, 6, and 7.	
receiving input files, wherein the input files are at least one web application graphical user interface;	As shown in step 510 in Figure 5 input files from a graphic designer are applied to the web application generator of the present invention. Fig. 5 also shows, at 535, how these input files are bound with other object programs to produce the source code which is sent to the requesting browser.	

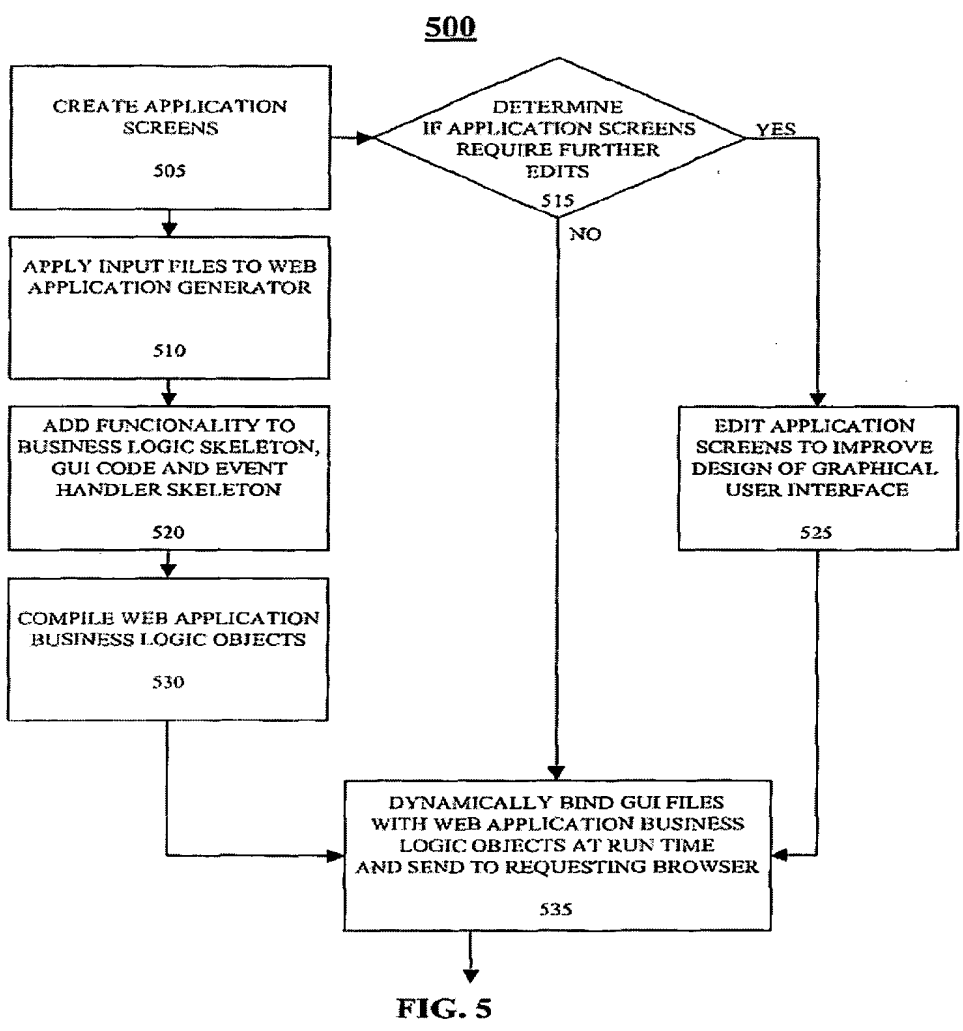


FIG. 5

generating an application framework code	Code 615 in Figure 6.
and an event handler skeleton	Code 620
wherein generating an event handler skeleton comprises: parsing at least one input file;	Step 720 in Figure 7
reviewing the parsed input file for one or more of a tag type, an attribute name and an attribute value; and	Step 725 in Fig. 7
determining an event handler method based on one or more of the tag type, the attribute name and the attribute value;	Step 730 in Figure 7

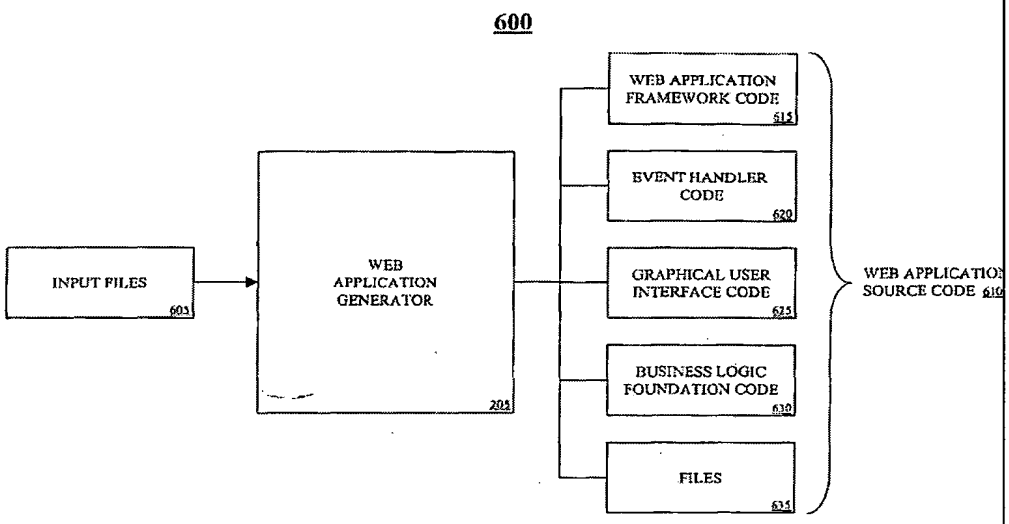
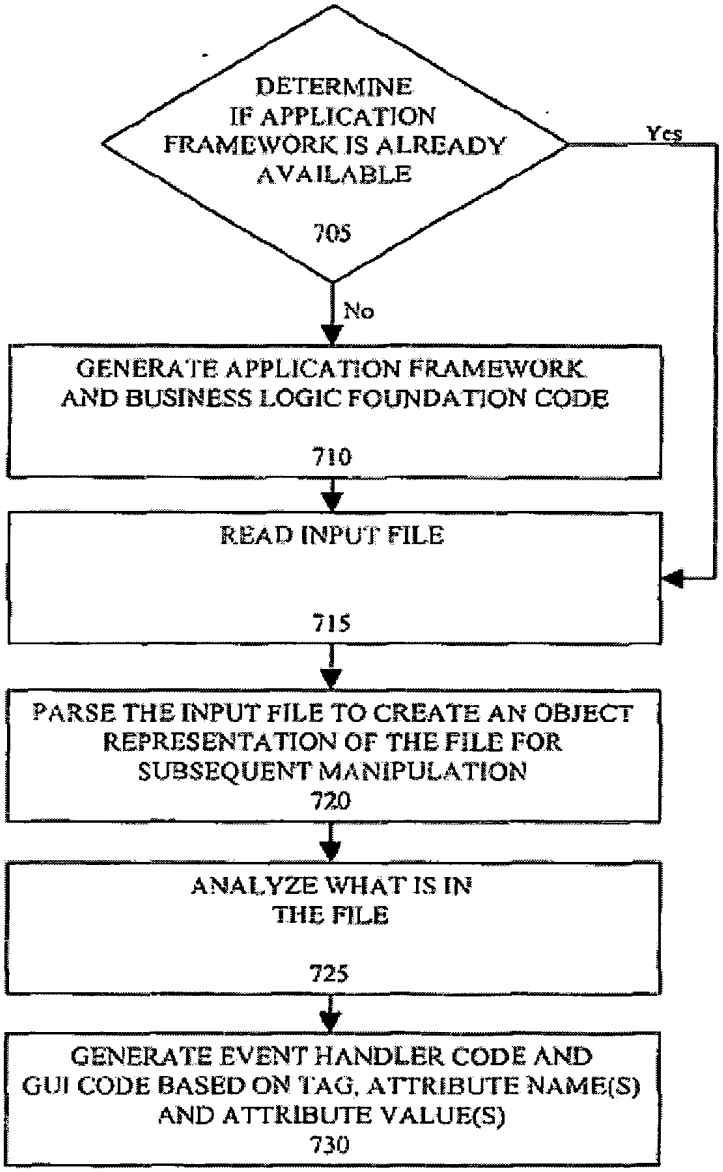


FIG. 6

receiving web application business logic objects;	The business logic objects are shown at 630 in Figure 6 and described on page 10 of the specification.
receiving event handler methods	Event handler code is shown at 620 in Figure 6.;
organizing the application framework code,	Application framework code is shown at 615 in Figure 6.
the web application business logic objects	Business logic objects are shown at 630 in Figure 6.
and the event handler methods	Event handler code is shown at 620 in Figure 6.
binding the web application source code with the input files at runtime	The foregoing codes are bound with input files 635 in Figure 6. This produces the web applications source wear. This is also shown at 535 in Figure 5.

Fig. 7



6. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Whether the rejections of claims 2, 4-29, 31-50, 52-64, 66-78, and 162-169 as unpatentable under 35 U.S.C. §103 (a) over Lau in view of Lindhorst et al. in further view of Quaeler-Bock et al. are improper.

7. ARGUMENT

Applicant has invented and claims a "method of generating computer code for a web application." The invention makes it easier for web developers to generate source code for a web application. In accordance with the invention, input files from graphic designers and source code from web developers are bound together at runtime. The input files are parsed to determine an event handler method based on tag type, attribute name and attribute value.

The Final Rejection is a combined-reference obviousness rejection in which the examiner has attempted to combine the disclosures of three unrelated references, Lau, Lindhorst et al., and Quaeler-Bock et al., **using the applicants' own teachings as a basis for making this combination**. We respectfully submit that this is improper hindsight reasoning and that the examiner has not made out a proper *prima facie* case of obviousness.

A. The Three Cited References Do Not Contain a Suggestion to Combine

1. Lau

The principal reference relied upon by the Examiner is U.S. Patent 5,987,247, Lau. The Lau patent relates to object oriented programming in general, but there is no mention of programming for a web application generator. More specifically, there is no mention of "parsing at least one input file," and "reviewing the parsed input file for one or more of a tag type, attribute name, and attribute value." (see applicants' claim 2). The Examiner concedes that this teaching is absent in Lau, but contends that this recitation is taught by U.S. Patent 6,337,696, Lindhorst et al.

The Examiner has also conceded the following with regard to other recitations of claim 2 (see pp. 3 – 4 of Final Rejection): "**Lau** did not explicitly state generating code for a *web* application."; **Lau** did not explicitly state *receiving event handler methods; and wherein generating an event handler skeleton further comprises: parsing at least one input file . . .; reviewing the parsed input file for a tag type, an attribute name and an attribute value; and determining an event handler method based on the tag type, the attribute name and the attribute value.*

2. Lindhorst et al.

Examiner relies on Lindhorst column 3, lines 37-40, but this part of the patent has nothing to do with parsing. Lindhorst does mention parsing at column 11, lines 47-49 with reference to Figure 5.

The only teaching of Lindhorst that is relevant is disclosure of the technique of parsing in computer programming in general. Column 13, lines 22-64 of Lindhorst, relied upon by Examiner, states:

> "The HTMP memory storage 220 also output events to the event pane 12 corresponding to events that are associated with scriptable HTMP tags."

Table 2, which follows, lists the scriptable HTML tags.

3. Quaeler-Bock et al

Finally the Office Action on pg. 5, first paragraph relies on Quaeler-Bock et al. for allegedly teaching that the implementation would have been obvious because one of ordinary skill in the art would have been motivated to reduce error-prone operations during code development. Column 3, lines 5-10 of Quaeler-Bock state:

> "Therefore, there is a need for a system and method for programming applications with GUIs that does not require the time-consuming and error-prone custom coding of GUI/internal variable synchronization routines, in general, and GUI'BO synchronization routines, in particular."

The foregoing teaching has almost no relevance to applicant's claimed invention, which includes parsing the input file, reviewing the parsed input file for a tag-type, an attribute name or an attribute value and "binding the application source code with the input files at runtime."

4. Summary of the Deficiencies of the Disclosures of the References

Lindhorst does not describe all the limitations of claim 2 regarding the generation of the event handler skeleton. In particular, Lindhorst describes parsing an input file (Col. 11, lines 41-44) and searching through the parsed document to send the separated object text, script text and HTML text from the parsed document into separate memory storage (col. 12, lines 61-64). However, Lindhorst does not describe "determining an event handler method based on the tag type, the attribute name and the attribute value." In Lindhorst the user links an event to a property using the graphical user interface described in Lindhorst. This linking determination is not based on a tag type, attribute name or attribute value of an object. In fact, Lindhorst states "when linking an event to a property, the user is attempting to use the event's trigger to change a property of the object." Applicant submits that changing a property of an object is different than determining an event handler method "based upon" a

tag or any attribute (i.e., property) of the object. None of the references contain a suggestion to combine the disclosures to achieve the advantages of Applicant's invention.

B. What Examiner Relies on as "Motivation" is the Unexpected Advantage of the Invention.

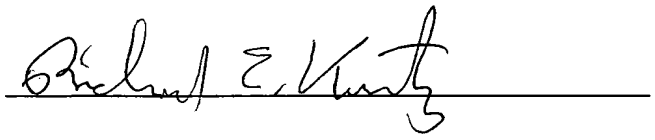
Starting at page 13 of the Final Rejection, Examiner argues that the advantages which applicant has provided are the motivation for combining the three references. Quite the contrary, the unexpected advantages of applicant's invention are objective evidence of invention. Unexpected results and the failure of others to achieve them are the classic objective evidence of invention. Graham v. John Deere Co. of Kansas City, et al., 148 U.S.P.Q. 459, 383 U.S. 1 (1966)

C No Prima Facie Case of Obviousness Because There is No Suggestion or Motivation to Combine

The unexpected advantages of applicant's invention cannot be used as the motivation to combine Lindhorst, Lau , and Quaeler-Bock with respect to how the generation of the event handler skeleton is performed as discussed above. According to MPEP 2143, "To establish a prima facie case of obviousness... first, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings." A prima facie case of obviousness has not been made because the Office Action did not indicate that there is any suggestion to implement the system of Lau with the generation of an event handler skeleton according to the particular way described by Lindhorst. Lacking a "suggestion" Examiner relies on the advantages of Applicant's invention as the motivation.

D. Conclusion

All of the claim limitation referred to above are in independent claims 27, 48, 64, 78 and 162-166. All of the independent claims are patentable for the reasons argued above. The remaining claims are dependent claims which are allowable for the same reason.



Date: January 31, 2007

Richard E. Kurtz
Registration No. 19,263

Woodcock Washburn LLP
Cira Centre
2929 Arch Street, 12th Floor
Philadelphia, PA 19104-2891
Telephone: (215) 568-3100
Facsimile: (215) 568-3439

8. CLAIMS APPENDIX

1. (Canceled)

2. (Rejected and On Appeal) A method of generating computer code for a web application, comprising:

receiving input files, wherein the input files are at least one web application graphical user interface;

generating an application framework code and an event handler skeleton, wherein generating an event handler skeleton comprises:

parsing at least one input file;

reviewing the parsed input file for one or more of a tag type, an attribute name and an attribute value; and

determining an event handler method based on one or more of the tag type, the attribute name and the attribute value;

receiving web application business logic objects;

receiving event handler methods;

organizing the application framework code, the web application business logic objects and the event handler methods into application source code; and

binding the web application source code with the input files at runtime.

3. (Canceled)

4. (Rejected and On Appeal) The method of claim 2, wherein the web application source code is generated in an object-oriented programming language.

5. (Rejected and On Appeal) The method of claim 4, wherein the object-oriented programming language is Java.

6. (Rejected and On Appeal) The method of claim 4, wherein the object-oriented programming language is C++.

7. (Rejected and On Appeal) The method of claim 2, further comprising determining if the application framework code is available for the web application.

8. (Rejected and On Appeal) The method of claim 2, further comprising generating a business logic foundation code.

9. (Rejected and On Appeal) The method of claim 2, further comprising generating a graphical user interface code.

10. (Rejected and On Appeal) The method of claim 9, wherein generating a graphical user interface code is based on the input files.

11. (Rejected and On Appeal) The method of claim 2, wherein generating an event handler skeleton is based on the input files.

12. (Rejected and On Appeal) The method of claim 2, further comprising compiling the web application source code.

13. (Rejected and On Appeal) The method of claim 2, further comprising interpreting the web application source code.

14. (Rejected and On Appeal) The method of claim 2, wherein the input files are in XML format.

15. (Rejected and On Appeal) The method of claim 2, wherein the input files are in HTML format.

16. (Rejected and On Appeal) The method of claim 2, wherein the input files are in cHTML format.

17. (Rejected and On Appeal) The method of claim 2, wherein the input files are in WML format.

18. (Rejected and On Appeal) The method of claim 2, further comprising receiving modified input files.

19. (Rejected and On Appeal) The method of claim 18, further comprising compiling the modified input files at runtime.

20. (Rejected and On Appeal) The method of claim 19, further comprising binding the web application source code with the compiled modified input files at runtime.

21. (Rejected and On Appeal) The method of claim 20, wherein the modified input files are compiled into DOM objects at runtime.

22. (Rejected and On Appeal) The method of claim 18, further comprising interpreting the modified input files at runtime.

23. (Rejected and On Appeal) The method of claim 22, further comprising binding the web application source code with the interpreted modified input files at runtime.

24. (Rejected and On Appeal) The method of claim 2, further comprising generating application runtime properties.

25. (Rejected and On Appeal) The method of claim 2, further comprising generating application SQL statements.

26. (Rejected and On Appeal) The method of claim 2, wherein the application framework code comprises an application object and a sen/let web application framework object.

27. (Rejected and On Appeal A method of generating computer code for a web application, comprising:

receiving in`put files, wherein the input files are at least one web application graphical user interface;

retrieving an application framework code from an application directory;

generating an event handler skeleton;

receiving web application business logic objects;

receiving event handler methods;

organizing the application framework, code, the web application, business logic objects and the event handler methods into application source code; and

binding the web application source code with the input files at runtime_and wherein generating an event handler skeleton further comprises:

parsing at least one input file;

reviewing the parsed input file for one or more of a tag type, an attribute name and an attribute value; and determining an event handler method based on the tag type, the attribute name and the attribute value.

28. (Rejected and On Appeal) The method of claim 27, further comprising retrieving a business logic foundation code.

29. (Rejected and On Appeal) The method of claim 27, further comprising generating a

business logic foundation code.

30. (Canceled)

31. (Rejected and On Appeal) The method of claim 27, wherein the web application source code is generated in an object-oriented programming language.

32. (Rejected and On Appeal) The method of claim 27, further comprising determining if the application framework code is available for the web application.

33. (Rejected and On Appeal) The method of claim 27, further comprising generating a graphical user interface code.

34. (Rejected and On Appeal) The method of claim 33, wherein generating a graphical user interface code is based on the input files.

35. (Rejected and On Appeal) The method of claim 27, wherein generating an event handler skeleton is based on the input files.

36. (Rejected and On Appeal) The method of claim 27, further comprising compiling the web application source code.

37. (Rejected and On Appeal) The method of claim 27, further comprising interpreting the web application source code.

38. (Rejected and On Appeal) The method of claim 27, wherein the input files are in XML format.

39. (Rejected and On Appeal) The method of claim 27, wherein the input files are in HTML format.

40. (Rejected and On Appeal) The method of claim 27, wherein the input files are in cHTML format.

41. (Rejected and On Appeal) The method of claim 27, wherein the input files are in WML format.

42. (Rejected and On Appeal) The method of claim 27, further comprising receiving

modified input files.

43. (Rejected and On Appeal) The method of claim 42, further comprising compiling the modified input files at runtime.

44. (Rejected and On Appeal l) The method of claim 43, further comprising binding the web application source code with the compiled modified input files at runtime.

45. (Rejected and On Appeal) The method of claim 42, further comprising interpreting the modified input files at runtime.

46. (Rejected and On Appeal) The method of claim 45, further comprising binding the web application source code with the interpreted modified input files at runtime.

47. (Rejected and On Appeal) The method of claim 27, wherein the application framework code comprises an application object and a servlet web application framework object.

48. (Rejected and On Appeal) method of generating computer code for a web application, comprising:

receiving input files, wherein the input files are at least one web application graphical user interface;

generating an application framework code and an event handler skeleton;

receiving web application business logic objects;

receiving event handler methods;

organizing the application framework code, the web application business logic objects and the event handler methods into web application source code;

receiving modified input files; and

binding the modified input files with the web application source code at runtime and wherein generating an event handler skeleton further comprises:

parsing at least one input file;

reviewing the parsed input file for one or more of a tag type, an attribute name and an attribute value; and

determining an event handler method based on the one or more of tag type, the attribute name and the attribute value.

49. (Rejected and On Appeal) The method of claim 48, further comprising compiling the modified input files at runtime.

50. (Rejected and On Appeal) The method of claim 48, further comprising interpreting the modified input files at runtime.

51. (Canceled)

52. (Rejected and On Appeal) The method of claim 48, wherein the web application source code is generated in an object-oriented programming language.

53. (Rejected and On Appeal) The method of claim 48, further comprising determining if the application framework code is available for the web application.

54. (Rejected and On Appeal) The method of claim 48, further comprising generating a business logic foundation code.

55. (Rejected and On Appeal) The method of claim 48, further comprising generating a graphical user interface code.

56. (Rejected and On Appeal) The method of claim 48, further comprising compiling the web application source code.

57. (Rejected and On Appeal) The method of claim 48, further comprising interpreting the web application source code.

58. (Rejected and On Appeal) The method of claim 48, wherein the input files are in XML format.

59. (Rejected and On Appeal) The method of claim 48, wherein the input files are in HTML format.

60. (Rejected and On Appeal) The method of claim 48, wherein the input files are in cHTML format.

61. (Rejected and On Appeal) The method of claim 48, wherein the input files are in WML format.

62. (Rejected and On Appeal) The method of claim 49, wherein the modified input files are compiled into DOM objects at runtime.

63. (Rejected and On Appeal) The method of claim 48, wherein the application framework code comprises an application object and a servlet web application framework object.

64. (Rejected and On Appeal)A method of generating computer code for a web application, comprising:

receiving input files, wherein the input files are at least one web application graphical user interface;

retrieving an application framework code from an application directory;

generating an event handler skeleton;

receiving web application business logic objects;

receiving event handler methods;

organizing the application framework code, the web application business logic objects and the event handler methods into web application source code;

and

binding modified input files with the web application source code at runtime and wherein generating an event handler skeleton further comprises:

parsing at least one input file;

reviewing the parsed input file for a tag type, an attribute name and an attribute value; and determining an event handler method based on the tag type, the attribute name and the attribute value.

65. (Canceled)

66. (Rejected and On Appeal) The method of claim 64, further comprising determining if the application framework code is available for the web application.

67. (Rejected and On Appeal) The method of claim 64, further comprising generating a business logic foundation code.

68. (Rejected and On Appeal) The method of claim 64, further comprising retrieving a business logic foundation code.

69. (Rejected and On Appeal) The method of claim 64, further comprising generating a

graphical user interface code.

70. (Rejected and On Appeal) The method of claim 64, wherein generating an event handler skeleton is based on the input files.

71. (Rejected and On Appeal) The method of claim 64, wherein the input files are in XML format.

72. (Rejected and On Appeal) The method of claim 64, wherein the input files are in HTML format.

73. (Rejected and On Appeal) The method of claim 64, wherein the input files are in HTML format.

74. (Rejected and On Appeal) The method of claim 64, wherein the input files are in WML format.

75. (Rejected and On Appeal) The method of claim 64, further comprising compiling the modified input files at runtime.

76. (Rejected and On Appeal) The method of claim 64, further comprising interpreting the modified input files at runtime.

77. (Rejected and On Appeal) The method of claim 64, wherein the application framework code comprises an application object and a servlet web application framework object.

78. (Rejected and On Appeal) method of generating computer code for a web application, comprising:

generating an event handler skeleton wherein the generating comprises parsing at least one input file, reviewing the parsed input file for one or more of a tag type, an attribute name and an attribute value, and determining an event handler method based on the one or more of tag type, the attribute name and the attribute value;

receiving a business logic foundation code, the event handler skeleton and a graphical user interface code; and

preparing web application business logic objects based on the business logic foundation code.

Claims 79-161 (canceled)

162. (Rejected and On Appeal) A device for generating computer code for a web application, comprising:

a storage device; and

a processor connected to the storage device, the storage device storing a program for controlling the processor;

the processor operative with the program to:

generate a business logic foundation code, an event handler skeleton and a graphical user interface code;

receive web application business logic objects from a web developer;

receive event handler methods from the web developer;

organize the application framework code, the web application business logic objects and the event handler methods into web application source code;

compile the web application source code;

compile the modified input files at runtime; and

bind the compiled modified input files with the compiled web application source code at runtime and wherein the processor is operative with the program in generating the event handler skeleton to:

parse at least one input file;

review the parsed input file for one or more of a tag type, an attribute name and an attribute value;

and

determine an event handler method based on the one or more of a tag type, the attribute name and the attribute value.

163. (Rejected and On Appeal) A device for generating computer code for a web application, comprising:

a storage device; and

a processor connected to the storage device, the storage device storing a program for controlling the processor;

the processor operative with the program to:

receive input files, wherein the input files are at least one web application graphical user interface;

generate an application framework code and an event handler skeleton; receive web application business logic objects; receive event handler methods;

organize the application framework code, the web application business logic objects and the event handler methods into application source code; and

bind the web application source code with the input files at runtime and wherein the processor is operative with the program in generating the event handler skeleton to:

parse at least one input file;

review the parsed input file for one or more of a tag type, an attribute name and an attribute value;

and

determine an event handler method based on the one or more of a tag type, the attribute name and the attribute value.

164. (Rejected and On Appeal) A device for generating computer code for a web application, comprising:

a storage device; and

a processor connected to the storage device, the storage device storing a program for controlling the processor;

the processor operative with the program to:

receive input files, wherein the input files are at least one web application graphical user interface;

retrieve an application framework code from an application directory;

generate an event handler skeleton;

receive web application business logic objects;

receive event handler methods;

organize the application framework code, the web application business logic objects and the event handler methods into application source code; and

bind the web application source code with the input files at runtime and wherein the processor is operative with the program in generating the event handler skeleton to:

parse at least one input file;

review the parsed input file for one or more of a tag type, an attribute name and an attribute value;

and

determine an event handler method based on the one or more of a tag type, the attribute name and the attribute value.

165. (Rejected and On Appeal) A device for generating computer code for a web application, comprising:

a storage device; and

a processor connected to the storage device, the storage device storing a program for controlling the processor;

the processor operative with the program to:

receive input files, wherein the input files are at least one web application graphical user interface;

generate an application framework code and an event handler skeleton; receive web application business logic objects; receive event handler methods;

organize the application framework code, the web application business logic objects and the event handler methods into web application source code; receive modified input files; and bind the modified input files with the web application source code at runtime and wherein the processor is operative with the program in generating the event handler skeleton to:

parse at least one input file;

review the parsed input file for one or more of a tag type, an attribute name and an attribute value;

and

determine an event handler method based on the one or more of a tag type, the attribute name and the attribute value.

166. (Rejected and On Appeal) A device for generating computer code for a web application, comprising:

a storage device; and

a processor connected to the storage device, the storage device storing a program for controlling the processor;

the processor operative with the program to:

receive input files, wherein the input files are at least one web application graphical user interface;

retrieve an application framework code from an application directory;

generate an event handler skeleton;

receive web application business logic objects;

receive event handler methods;

organize the application framework code, the web application business logic objects and the event handler methods into web application source code;

receive modified input files; and

bind the modified input files with the web application source code at runtime and wherein the processor is operative with the program in generating the event handler skeleton to:

parse at least one input file;

review the parsed input file for one or more of a tag type, an attribute name and an attribute value;

and

determine an event handler method based on the one or more of a tag type, the attribute name and the attribute value.

167. (Rejected and On Appeal) The method of claim 2, further comprising:

determining if an application framework code is available for the web application; and if the application framework code is not available, then generating the application framework code.

168. (Rejected and On Appeal) The device of claim 166, further comprising:

a determining mechanism configured to determine if an application framework code is available for the web application; and

a code generator configured to generate the application framework code.

169. (Rejected and On Appeal) The device of claim 162, wherein the processor is further operative with the program to:

determine if an application framework code is available for the web application; and if the application framework code is not available, then generate the application framework code.

170. (Canceled)

171. (Canceled)

172. (Canceled)

173. (Canceled)

174. (Canceled)